Wind Bells and the Lonely Flute

For Computer Generated Tape and Flute

Dur. 2 min 35 sec

by

Ali Riza SARAL

...

April 2024

Copyright notice

Performance Restricted Attribution-NonCommercial-NoDerivs 3.0

Please send a digital recording of your performance
to the composer at arsaral(AATT)yahoo.com

The piece is written to be used for good causes.

CONTENTS

Introduction

Score

FORWORD

I have composed Windbells and the Lonely Flute as a supplementary to my studies under the auspieces of Manolis Ekmektsioglou, at MİAM (Institute for Advanced Research in Music – Istanbul Technical University).

Windbells and the Lonely Flute is not an isolated work in my portfolio; rather, it is part of a continuum of compositions in computer-aided music. The following is a list of these works:

2016, Dec – At The Edge for Cello solo

2020, Apr - Piece for Brass Quintet and Strings

2020, Jul – Algorithmic Sketches for Piano solo

2020, Aug – Algorithms for Synthetic Piano

2020, Aug – Sinusoidal Etudes for Piano solo

2021, Jun – Song of the Windschimes for computer (JFugue)

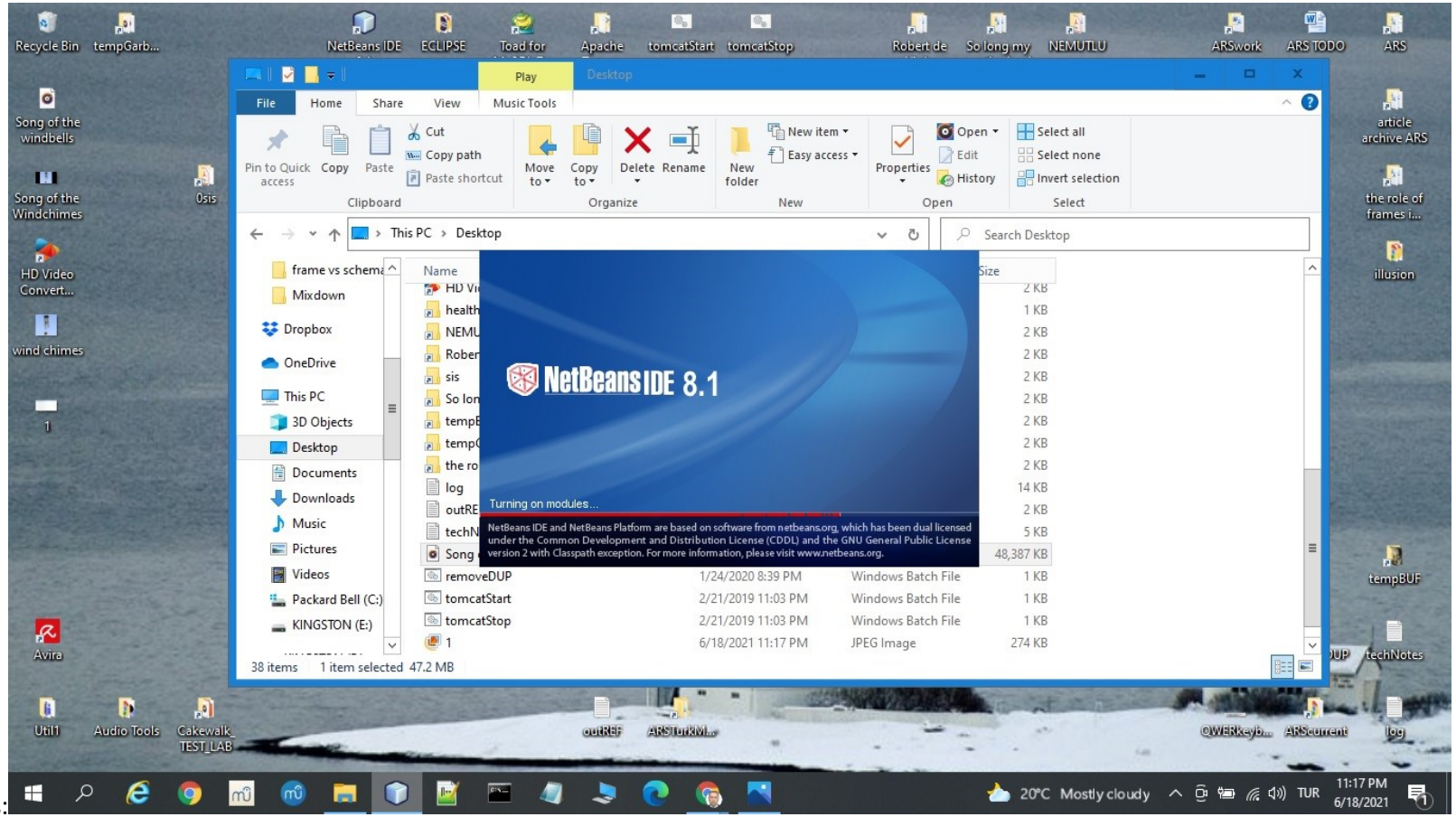
2024, Apr – Song of the Windbells and the Lonely Flute for computer (cSound)

The first five works, up to and including 2020, utilized JAVA, JFugue, MuseSCORE, and CakeWalk. These tools were employed to create MIDI scores, which were then imported into MuseSCORE as conventional music scores. The final edits were made using CakeWalk. JAVA and JFugue were utilized to generate random algorithms within the framework of limited aleatory.

The last two works, from 2021 onwards, employ frequencies instead of traditional pitches, thus falling into the category of electronic music pieces. They utilize similar computer algorithms with limited aleatory features. The latter piece also incorporates an acoustic flute with microtones. It builds upon material from the 2021 Windchimes work, enhancing it with cSound. Additionally, the final piece incorporates additive synthesis – RISSET – towards its conclusion.

INTRODUCTION

1- Song of the Wind Schimes is composed by using JAVA on NetBeans and synthesized with Jfugue and edited on Studio One. The process can be followed as:



microTonalTST - NetBeans IDE 8.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

skyTST1.java | Bagatelle.java | SIS.java | ARSMicroTonalFuncs.java | timerExample.java

Source History

```
55          //buf=performClimbingLineWithConstantSpaces(buf,40,100, 1200, "q", "i");
56          //buf=performClimbingLineWithRandomdureeWithConstantSpaces(buf,40,100, 1200, 1, 6, "q");
57          //buf=performClimbingLineWithRandomdureeWithRandomSpaces(buf,40,100, 1200, 1, 6, 2, 4);
58
59          //buf= performDescendingLine(buf,100,100,12000,"q");
60          //buf=performDescendingLineWithRandomduree(buf,40,100, 12000, 1, 6);
61          //buf=performDescendingLineWithConstantSpaces(buf,40,100, 12000, "i","q");
62          //buf=performDescendingLineWithRandomdureeWithConstantSpaces(buf,400,50, 16000, 1, 6, "q");
63          //buf=performDescendingLineWithRandomdureeWithRandomSpaces(buf,40,100, 12000, 1, 6, 2, 4);
64  //-----------------------------------------------
65          //buf=performRandomClimbingLine(buf,100,160,0,"q",500);
66          //buf=performRandomClimbingLineWithConstantSpaces(buf,100,160,0,"q",500,"h");
67          //buf=performRandomClimbingLineWithRandomDuree(buf,40,100, 160, 1, 6, 500);
68          //buf=performRandomClimbingLineWithRandomDureeWithConstantSpaces(buf,400,50, 2000, 1, 6, "q",500);
69              //performRandomClimbingLineWithRandomDureeWithConstantSpaces(String buf,int step,int stepCount, int begFreq, int minDuree, int maxDuree, String spaceDuree, int rando
70          //buf=performRandomClimbingLineWithRandomDureeWithRandomSpaces(buf,100,50, 160, 1, 6, 2, 4,200);
71
72          //buf=performRandomDescendingLine(buf,100,160,12000,"q",500);
73  // buf=performRandomDescendingLineWithConstantSpaces(buf,100,160,12000,"s",500,"h");
74          //buf=performRandomDescendingLineWithRandomDuree(buf,40,100, 12000, 1, 6, 500);
75          //buf=performRandomDescendingLineWithRandomDureeWithConstantSpaces(buf,400,50, 12000, 1, 6, "q",500);
76          //buf=performRandomDescendingLineWithRandomDureeWithRandomSpaces(buf,200,100, 12000, 1, 6, 2, 4,500);
77  //-----------------------------------------------
78            //buf=performSinus(buf,32,200, 1000,"i",6000);  // i
79            //buf=performSinusWithConstantSpaces(buf,17,200, 1000,"s","i",5000);
80            //buf=performSinusWithRandomDuree(buf,17,200, 1500, 3, 5,7000);
81            buf=performSinusWithRandomDureeWithConstantSpaces(buf,33,10, 1500, 4, 5,"s",7000);   //20'
82            //buf=performSinusWithRandomDureeWithRandomSpaces(buf,17,200, 1000, 3, 5, 1,2,6000);     //20'
83
84            //buf=performClimbingSinus(buf,14,200, 120, 500,"s");
85  //buf=performClimbingSinusWithConstantSpaces(buf,17,200, 12000, 500,"s","i");
86            //buf=performClimbingSinusWithRandomDuree(buf,17,200, 120,500, 1, 3);
```

Output | Search Results | Usages

1:1 INS

20°C Mostly cloudy TUR 11:16 PM 6/18/2021

microTonalTST - NetBeans IDE 8.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

skyTST1.java | Bagatelle.java | SIS.java | ARSMicroTonalFuncs.java | timerExample.java

Source | History

```
                buffy.append(getDuree(duree)+" ");
                buf= buf +" "+ buffy.toString();// buf="";

                System.out.println(buf);
                }
            return buf;
        }
        public static String performSinusWithRandomDureeWithConstantSpaces(String buf,int step,int stepCount, int freqDepthAmplitude, int minDuree, int maxDuree, String spaceDuree,
            double angle=0, angleRadian=0, anglePrev=0;
            for (int i=0;i<stepCount;i++){
                angle= (i*step) % 360;
                angleRadian = Math.toRadians(angle);

                StringBuilder buffy = new StringBuilder();
                buffy.append("m");
                buffy.append(refFreq+(freqDepthAmplitude * Math.sin(angleRadian)));
                        System.out.println(angle + " = "+ Math.sin(angleRadian) +"---->"+ (freqDepthAmplitude * Math.sin(angleRadian)));
                buf= buf +" "+ buffy.toString();// buf="";
                buf=buf + " R"+spaceDuree;

                System.out.println(buf);
                }
            return buf;
        }
        public static String performSinusWithRandomDureeWithRandomSpaces(String buf,int step,int stepCount, int freqDepthAmplitude, int minDuree, int maxDuree, int minSpace, int ma
            double angle=0, angleRadian=0, anglePrev=0;
            int duree=0;
            int space=0;
            int isSpace=0;
            for (int i=0;i<stepCount;i++){
                angle= (i*step) % 360;
                angleRadian = Math.toRadians(angle);
```

Output | Search Results | Usages

524:6 INS

20°C Mostly cloudy TUR 11:18 PM 6/18/2021

microTonalTST - NetBeans IDE 8.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

skyTST1.java | Bagatelle.java | SIS.java | ARSMicroTonalFuncs.java | timerExample.java

Source | History

```
//      buf += "T150 V2 :CON(7, 30) ";
//     buf+=performRandomfreqBlock(buf,13,3500, 5000, "q" );
//=====================================================
//      //IIA1-----------------------------------
//        buf = "T250 V0 :CON(7, 80) ";
//        buf=performRandomfreqBlockWithRandomdureeWithRandomSpaces(buf,550,4500, 5500, 4, 6,2 ,6);
//      //IIA2-----------------------------------
//        buf = "T250 V0 :CON(7, 60) ";
//        buf=performRandomfreqBlockWithRandomdureeWithRandomSpaces(buf,150,4500, 5500, 2, 5,1 ,3);
//      //IIIA1----------------------------------
//        buf = "T250 V0 :CON(7, 90) ";
//        buf=performRandomClimbingLineWithRandomDureeWithConstantSpaces(buf,100,50, 2000, 2, 5, "q",20);
//-------------------------------------------------------
//     //IB1--------------------
//      buf = "T150 V2 :CON(7, 30) ";
//     buf+=performRandomfreqBlock(buf,13,3500, 5000, "q" );
//     buf+=performRandomfreqBlock(buf,13,3500, 5000, "i" );
//     buf+=performRandomfreqBlock(buf,13,3500, 5000, "s" );
//     buf+=performRandomfreqBlock(buf,100,3500, 5000, "t" );
//-------------------------------------------------------
//      //IIC1
//        buf = "T250 V0 :CON(7, 60) ";
//        buf=performSinusWithRandomDureeWithRandomSpaces(buf,17,200, 2500, 3, 5, 3,4,10000);
        //IIC2
        buf = "T250 V0 :CON(7, 70) ";
        buf=performSinusWithRandomDureeWithRandomSpaces(buf,17,480, 1000, 4, 5, 5,6,12000);
//-------------------------------------------------------
//      //ID1
//        buf = "T250 V0 :CON(7, 90) ";
//        buf=performClimbingSinusWithRandomDuree(buf,17,300, 5000,1500, 3, 4);
//        buf+=performClimbingSinusWithRandomDuree(buf,170,30, 7000,300, 3, 4);
//        buf+=performClimbingSinusWithRandomDuree(buf,170,10, 9000,700, 3, 4);
//        buf+=performClimbingSinusWithRandomDuree(buf,170,30, 12000,1000, 3, 4);
//      //ID2
```

Output | Search Results | Usages

156:14 INS

20°C Mostly cloudy TUR 11:19 PM 6/18/2021

microTonalTST - NetBeans IDE 8.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

skyTST1.java | Bagatelle.java | SIS.java | ARSMicroTonalFuncs.java | timerExample.java

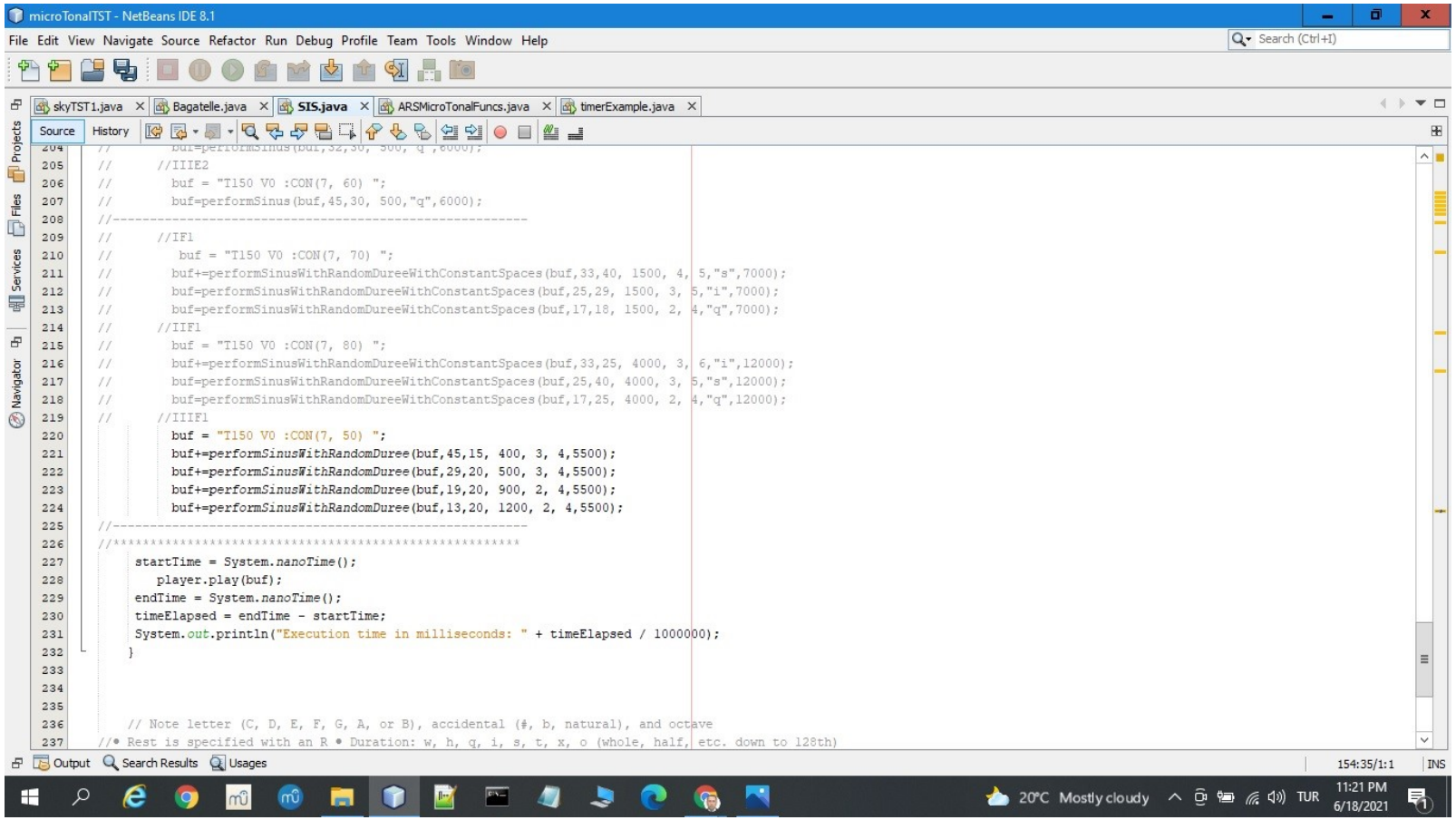
Source | History

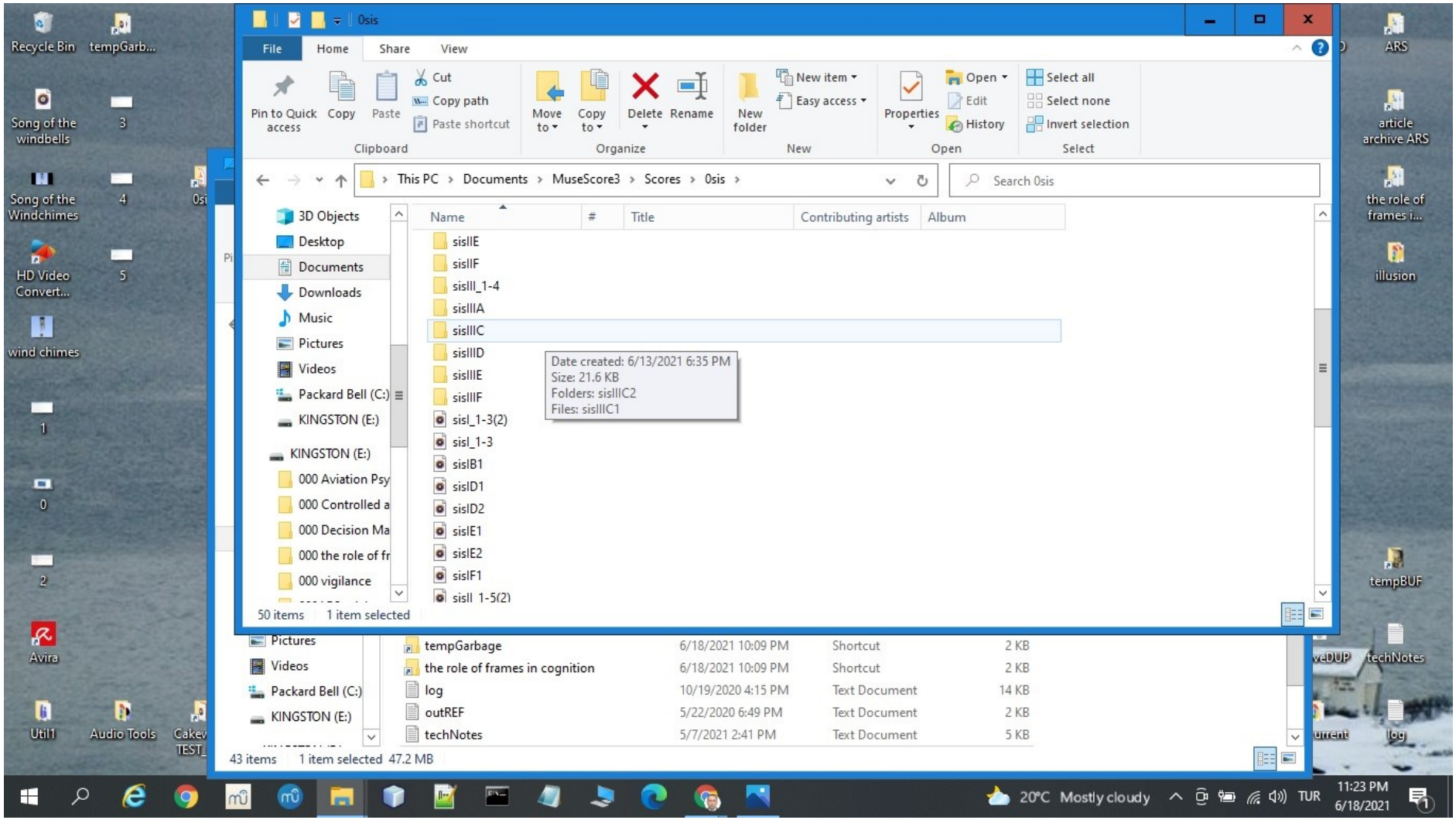
```java
import static microtonaltst.ARSMicroTonalFuncs.performSinusWithRandomDureeWithConstantSpaces;
import static microtonaltst.ARSMicroTonalFuncs.performSinusWithRandomDureeWithRandomSpaces;
import org.jfugue.midi.MidiFileManager;
import org.jfugue.midi.MidiParser;
import org.staccato.StaccatoParserListener;
/**
 *
 * @author ars
 */
public class SIS {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws InvalidMidiDataException, IOException {
        MidiParser parser = new MidiParser();
        StaccatoParserListener listener = new StaccatoParserListener();
        parser.addParserListener(listener);
       // parser.parse(MidiSystem.getSequence(new File("C:\\Users\\ars\\Documents\\MuseScore3\\Scores\\0sis\\sisTST0.mid")));



        Player player = new Player();
        Pattern pattern = new Pattern();


        try {
            MidiFileManager
                    .savePatternToMidi(pattern, new File("C:\\\\Users\\\\ars\\\\Documents\\\\MuseScore3\\\\Scores\\\\0sis\\\\sisTST0.mid"));
        } catch (IOException ex) {
        }
//        pattern.add(" T10 V0 L0 I40 :CON(100,0) :CON(101,0) :CON(6,1) C R3 L1");
//        int PW = 8192; // Default Pitch Bend
```

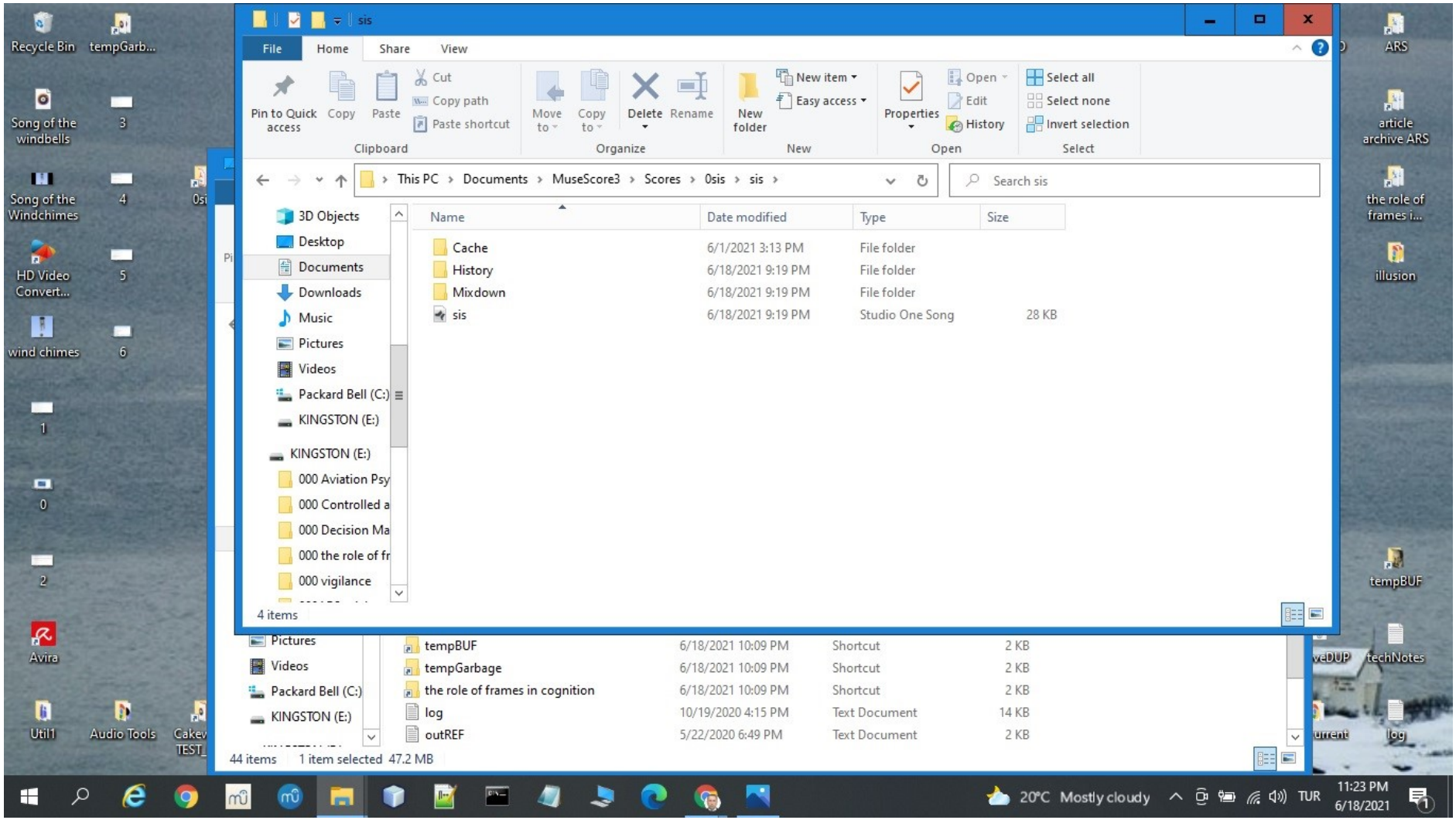
Output | Search Results | Usages

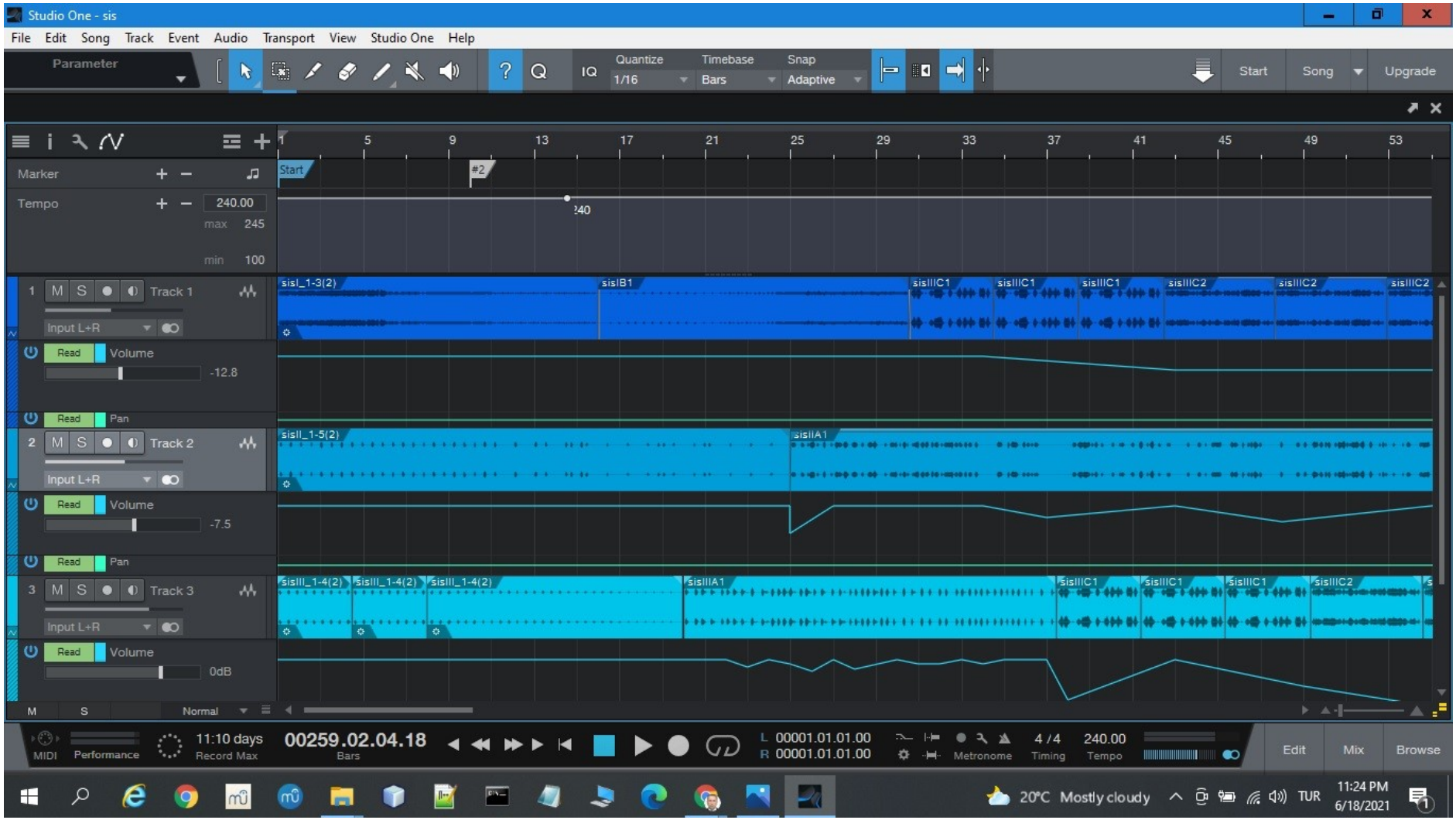
154:35/1:1 | INS

20°C Mostly cloudy TUR 11:20 PM 6/18/2021


microTonalTST - NetBeans IDE 8.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
skyTST1.java
Bagatelle.java
SIS.java
ARSMicroTonalFuncs.java
timerExample.java
Source
History
Projects
Files
Services
Navigator
//IIIE2
buf = "T150 V0 :CON(7, 60) ";
buf=performSinus(buf,45,30, 500,"q",6000);
//IF1
buf = "T150 V0 :CON(7, 70) ";
buf+=performSinusWithRandomDureeWithConstantSpaces(buf,33,40, 1500, 4, 5,"s",7000);
buf=performSinusWithRandomDureeWithConstantSpaces(buf,25,29, 1500, 3, 5,"i",7000);
buf=performSinusWithRandomDureeWithConstantSpaces(buf,17,18, 1500, 2, 4,"q",7000);
//IIF1
buf = "T150 V0 :CON(7, 80) ";
buf+=performSinusWithRandomDureeWithConstantSpaces(buf,33,25, 4000, 3, 6,"i",12000);
buf=performSinusWithRandomDureeWithConstantSpaces(buf,25,40, 4000, 3, 5,"s",12000);
buf=performSinusWithRandomDureeWithConstantSpaces(buf,17,25, 4000, 2, 4,"q",12000);
//IIIF1
buf = "T150 V0 :CON(7, 50) ";
buf+=performSinusWithRandomDuree(buf,45,15, 400, 3, 4,5500);
buf+=performSinusWithRandomDuree(buf,29,20, 500, 3, 4,5500);
buf+=performSinusWithRandomDuree(buf,19,20, 900, 2, 4,5500);
buf+=performSinusWithRandomDuree(buf,13,20, 1200, 2, 4,5500);
startTime = System.nanoTime();
player.play(buf);
endTime = System.nanoTime();
timeElapsed = endTime - startTime;
System.out.println("Execution time in milliseconds: " + timeElapsed / 1000000);
}
// Note letter (C, D, E, F, G, A, or B), accidental (#, b, natural), and octave
//• Rest is specified with an R • Duration: w, h, q, i, s, t, x, o (whole, half, etc. down to 128th)
Output
Search Results
Usages
154:35/1:1
INS
20°C Mostly cloudy
TUR
11:21 PM
6/18/2021


0sis
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New item
Easy access
New
Properties
Open
Edit
History
Select all
Select none
Invert selection
Select
This PC › Documents › MuseScore3 › Scores › 0sis ›
Search 0sis
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Packard Bell (C:)
KINGSTON (E:)
KINGSTON (E:)
000 Aviation Psy
000 Controlled a
000 Decision Ma
000 the role of fr
000 vigilance
Name
#
Title
Contributing artists
Album
sislIE
sislIF
sislII_1-4
sislIIA
sislIIC
sislIID
sislIIE
sislIIF
sisl_1-3(2)
sisl_1-3
sislB1
sislD1
sislD2
sislE1
sislE2
sislF1
sisll_1-5(2)
Date created: 6/13/2021 6:35 PM
Size: 21.6 KB
Folders: sislIIC2
Files: sislIIC1
50 items
1 item selected
Pictures
Videos
Packard Bell (C:)
KINGSTON (E:)
tempGarbage 6/18/2021 10:09 PM Shortcut 2 KB
the role of frames in cognition 6/18/2021 10:09 PM Shortcut 2 KB
log 10/19/2020 4:15 PM Text Document 14 KB
outREF 5/22/2020 6:49 PM Text Document 2 KB
techNotes 5/7/2021 2:41 PM Text Document 5 KB
43 items
1 item selected 47.2 MB
20°C Mostly cloudy
TUR
11:23 PM
6/18/2021


sis
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New item
Easy access
New
Properties
Open
Edit
History
Select all
Select none
Invert selection
Select
This PC › Documents › MuseScore3 › Scores › 0sis › sis
Search sis
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Packard Bell (C:)
KINGSTON (E:)
KINGSTON (E:)
000 Aviation Psy
000 Controlled a
000 Decision Ma
000 the role of fr
000 vigilance
Name
Date modified
Type
Size
Cache 6/1/2021 3:13 PM File folder
History 6/18/2021 9:19 PM File folder
Mixdown 6/18/2021 9:19 PM File folder
sis 6/18/2021 9:19 PM Studio One Song 28 KB
4 items
tempBUF 6/18/2021 10:09 PM Shortcut 2 KB
tempGarbage 6/18/2021 10:09 PM Shortcut 2 KB
the role of frames in cognition 6/18/2021 10:09 PM Shortcut 2 KB
log 10/19/2020 4:15 PM Text Document 14 KB
outREF 5/22/2020 6:49 PM Text Document 2 KB
44 items
1 item selected 47.2 MB
20°C Mostly cloudy
TUR
11:23 PM
6/18/2021


Studio One - sis
File Edit Song Track Event Audio Transport View Studio One Help
Parameter
Quantize 1/16
Timebase Bars
Snap Adaptive
Start
Song
Upgrade
Marker
Tempo 240.00
max 245
min 100
Track 1
Input L+R
Read Volume -12.8
Read Pan
Track 2
Input L+R
Read Volume -7.5
Read Pan
Track 3
Input L+R
Read Volume 0dB
Normal
MIDI
Performance
11:10 days
Record Max
00259.02.04.18
Bars
L 00001.01.01.00
R 00001.01.01.00
Metronome
4 / 4
Timing
240.00
Tempo
Edit
Mix
Browse
20°C Mostly cloudy
TUR
11:24 PM
6/18/2021

1- Source code base for Wind chimes

```
/*
 * Ali Rıza SARAL
 */
package microtonaltst;
import java.io.File;
import java.io.IOException;
import org.jfugue.pattern.Pattern;
import org.jfugue.player.Player;
import java.util.Random;
import javax.sound.midi.InvalidMidiDataException;
import javax.sound.midi.MidiSystem;
import static microtonaltst.ARSMicroTonalFuncs.performClimbingSinusWithConstantSpaces;
import static microtonaltst.ARSMicroTonalFuncs.performClimbingSinusWithRandomDuree;
import static microtonaltst.ARSMicroTonalFuncs.performClimbingSinusWithRandomDureeWithConstantSpaces;
import static microtonaltst.ARSMicroTonalFuncs.performRandomClimbingLineWithRandomDureeWithConstantSpaces;
import static microtonaltst.ARSMicroTonalFuncs.performRandomfreqBlock;
import static microtonaltst.ARSMicroTonalFuncs.performRandomfreqBlockWithRandomduree;
import static microtonaltst.ARSMicroTonalFuncs.performRandomfreqBlockWithRandomdureeWithRandomSpaces;
import static microtonaltst.ARSMicroTonalFuncs.performSinus;
import static microtonaltst.ARSMicroTonalFuncs.performSinusWithConstantSpaces;
import static microtonaltst.ARSMicroTonalFuncs.performSinusWithRandomDuree;
import static microtonaltst.ARSMicroTonalFuncs.performSinusWithRandomDureeWithConstantSpaces;
import static microtonaltst.ARSMicroTonalFuncs.performSinusWithRandomDureeWithRandomSpaces;
import org.jfugue.midi.MidiFileManager;
import org.jfugue.midi.MidiParser;
import org.staccato.StaccatoParserListener;
/**
 *
 * @author ars
 */
public class SIS {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws InvalidMidiDataException, IOException {
        MidiParser parser = new MidiParser();
```

```
        StaccatoParserListener listener = new StaccatoParserListener();
        parser.addParserListener(listener);
       // parser.parse(MidiSystem.getSequence(new File("C:\\Users\\ars\\Documents\\MuseScore3\\Scores\\0sis\\sisTST0.mid")));




        Player player = new Player();
        Pattern pattern = new Pattern();


        try {
            MidiFileManager
                    .savePatternToMidi(pattern, new File("C:\\\\Users\\\\ars\\\\Documents\\\\MuseScore3\\\\Scores\\\\0sis\\\\sisTST0.mid"));
        } catch (IOException ex) {
        }
//        pattern.add(" T10 V0 L0 I40 :CON(100,0) :CON(101,0) :CON(6,1) C R3 L1");
//        int PW = 8192; // Default Pitch Bend
//        int steps=250; // Let us consider 250 steps in one semitone
//        double duration = 0.25/steps; // Duration of each step
//        int PWValue=8192/steps; // This is the value of pitch bend to be changed gradually
//        for(int i=0;i<steps;i++) // S1 to R1
//        {
//        PW=PW+PWValue;
//
//        pattern.add(" R/"+duration +" :PW("+PW+") ");
//        System.out.println(pattern.toString());
//        }
//          player.play(pattern);
        // player.play("T10 m512.3q m515.3q m520.3q m525.3q m530.3q m535.3q m540.3q  ");
        //player.play("T10 m612.6q m275.6q m279.6q m290.6q  m294.3q ");
         //player.play("T10 m440q m445q m450q m455q  m460q ");
       // player.play("T10 :PitchWheel(5192) 72q :PitchWheel(8192) :PitchWheel(4000) 72q :PitchWheel(8192) :PitchWheel(2500) 72q
:PitchWheel(8192) ");
       //player.play("T10 A4q  :PitchWheel(8100) A4q  :PitchWheel(5500) A4q  :PitchWheel(3000) A4q   ");

       String buf="";
//       for (int i=0;i<160;i++){
//           dev=i*100;
```

```
//          StringBuilder buffy = new StringBuilder();
//       buffy.append("m");
//       buffy.append((int)dev);
//       buffy.append("q ");
//      buf= buf +" "+ buffy.toString();
//      System.out.println(buf);
//      }
     //buf= performClimbingLine(buf,100,160,0);
     //buf= performClimbingLine(buf,10,10,12000);
     //buf= performDescendingLine(buf,100,100,12000);
     //buf=performRandomfreqBlock(buf,40,12000, 10000 );
    //buf=performRandomfreqBlockWithRandomduree(buf,40,1000, 12000, 1, 6);
    //buf=performRandomfreqBlockWithRandomdureeWithRandomSpaces(buf,40,1000, 12000, 1, 6,2 ,4);
    //I0--------------------
    long startTime=0,endTime=0,timeElapsed=0;
//     //I1--------------------
//     buf = "T300 V0 :CON(7, 80) ";
//     buf+=performRandomfreqBlock(buf,50,9000, 11000, "i" );
//     //I2---------------------
//     buf += "T300 V0 :CON(7, 65) ";
//     buf+=performRandomfreqBlock(buf,50,7500, 9000, "i" );
//     //I3---------------------
//     buf += "T250 V0 :CON(7, 50) ";
//     buf+=performRandomfreqBlock(buf,42,6000, 7500, "i" );
//     //==========================
//     //II1--------------------
//      buf = "T250 V1 :CON(7, 80) ";
//     buf+=performRandomfreqBlock(buf,10,5500, 6000, "h" );
//    //II2--------------------
//      buf += "T250 V1 :CON(7, 80) ";
//     buf+=performRandomfreqBlock(buf,10,5500, 6000, "h" );
//     //II3--------------------
//      buf += "T250 V1 :CON(7, 80) ";
//     buf=performRandomfreqBlockWithRandomduree(buf,8,5500, 6000, 0, 2);
//    //II4--------------------
//      buf += "T225 V1 :CON(7, 65) ";
//     buf=performRandomfreqBlockWithRandomduree(buf,9,5000, 5500, 0, 2);
//    //II5--------------------
//      buf += "T200 V1 :CON(7, 65) ";
```

```
//     buf=performRandomfreqBlockWithRandomduree(buf,10,4500, 5500, 0, 2);
//     //==========================
//     //III1--------------------
//      buf = "T150 V2 :CON(7, 65) ";
//     buf+=performRandomfreqBlock(buf,13,3500, 5000, "q" );
//     //III2--------------------
//      buf += "T150 V2 :CON(7, 65) ";
//     buf+=performRandomfreqBlock(buf,12,3500, 5000, "q" );
//     //III3--------------------
//      buf += "T150 V2 :CON(7, 50) ";
//     buf+=performRandomfreqBlock(buf,13,3500, 5000, "q" );
//     //III4--------------------
//      buf += "T150 V2 :CON(7, 30) ";
//     buf+=performRandomfreqBlock(buf,13,3500, 5000, "q" );
//=====================================================
//      //IIA1-----------------------------------
//        buf = "T250 V0 :CON(7, 80) ";
//        buf=performRandomfreqBlockWithRandomdureeWithRandomSpaces(buf,550,4500, 5500, 4, 6,2 ,6);
//      //IIA2-----------------------------------
//        buf = "T250 V0 :CON(7, 60) ";
//        buf=performRandomfreqBlockWithRandomdureeWithRandomSpaces(buf,150,4500, 5500, 2, 5,1 ,3);
//      //IIIA1----------------------------------
//        buf = "T250 V0 :CON(7, 90) ";
//        buf=performRandomClimbingLineWithRandomDureeWithConstantSpaces(buf,100,50, 2000, 2, 5, "q",20);
//-------------------------------------------------------
     //IB1--------------------
      buf = "T150 V2 :CON(7, 80) ";
     buf+=performRandomfreqBlock(buf,13,3500, 5000, "q" );
     buf+=performRandomfreqBlock(buf,13,3500, 5000, "i" );
     buf+=performRandomfreqBlock(buf,13,3500, 5000, "s" );
     buf+=performRandomfreqBlock(buf,100,3500, 5000, "t" );
//-------------------------------------------------------
////      //IIC1
////        buf = "T250 V0 :CON(7, 60) ";
////        buf=performSinusWithRandomDureeWithRandomSpaces(buf,17,200, 2500, 3, 5, 3,4,10000);
//      //IIC2
//        buf = "T120 V0 :CON(7, 70) ";
//        buf=performSinusWithRandomDureeWithRandomSpaces(buf,17,480, 1000, 4, 5, 5,6,12000);
//-------------------------------------------------------
```

```
//      //ID1
//        buf = "T250 V0 :CON(7, 90) ";
//        buf=performClimbingSinusWithRandomDuree(buf,17,300, 5000,1500, 3, 4);
//        buf+=performClimbingSinusWithRandomDuree(buf,170,30, 7000,300, 3, 4);
//        buf+=performClimbingSinusWithRandomDuree(buf,170,10, 9000,700, 3, 4);
//        buf+=performClimbingSinusWithRandomDuree(buf,170,30, 12000,1000, 3, 4);
//      //ID2
//            buf = "T250 V0 :CON(7, 100) ";
//            buf=performClimbingSinusWithRandomDuree(buf,17,400, 10000,2000, 3, 4);
          //        buf+=performClimbingSinusWithRandomDuree(buf,17,40, 10000,100, 3, 4);
          //        buf+=performClimbingSinusWithRandomDuree(buf,17,40, 10000,100, 3, 4);
//      //IID1
//        buf = "T250 V0 :CON(7, 100) ";
//        buf=performClimbingSinusWithRandomDuree(buf,16,200, 8000,2500, 4, 5);
          //        buf+=performClimbingSinusWithRandomDuree(buf,16,100, 8000,250, 4, 5);
          //        buf+=performClimbingSinusWithRandomDuree(buf,16,100, 8000,250, 4, 5);
//      //IID2
//            buf = "T200 V0 :CON(7, 80) ";
//            buf=performClimbingSinusWithRandomDuree(buf,16,250, 7000,1500, 3, 5);
          //        buf+=performClimbingSinusWithRandomDuree(buf,16,100, 7000,400, 4, 5);
          //        buf+=performClimbingSinusWithRandomDuree(buf,16,100, 7000,400, 4, 5);
//      //IIID1
//        buf = "T150 V0 :CON(7, 80) ";
//        buf=performClimbingSinusWithRandomDuree(buf,16,400, 6000,500, 4, 6);
//        buf=performClimbingSinusWithRandomDuree(buf,16,300, 6000,500, 5, 6);
//        buf=performClimbingSinusWithRandomDuree(buf,16,300, 6000,500, 5, 6);
//        buf=performClimbingSinusWithRandomDuree(buf,16,300, 6000,500, 5, 6);
//-------------------------------------------------------------------------------
```

```
//      //IE1
//        buf = "T150 V0 :CON(7, 50) ";
//        buf=performSinus(buf,32,60, 1500,"i",7000);
//      //IE2
//        buf = "T150 V0 :CON(7, 55) ";
```

```
//        buf=performSinusWithRandomDuree(buf,17,110, 1500, 3, 5,7000);
//      //IIE1
//        buf = "T150 V0 :CON(7, 40) ";
//        buf=performSinus(buf,17,120, 4000,"s",12000);
//      //IIE2
//          buf = "T150 V0 :CON(7, 50) ";
//          buf=performSinusWithRandomDuree(buf,17,120, 4000, 3, 6,12000);
//      //IIIE1
//        buf = "T150 V0 :CON(7, 60) ";
//        buf=performSinus(buf,32,30, 500,"q",6000);
//      //IIIE2
//        buf = "T150 V0 :CON(7, 60) ";
//        buf=performSinus(buf,45,30, 500,"q",6000);
//-------------------------------------------------------
//      //IF1
//         buf = "T150 V0 :CON(7, 70) ";
//        buf+=performSinusWithRandomDureeWithConstantSpaces(buf,33,40, 1500, 4, 5,"s",7000);
//        buf=performSinusWithRandomDureeWithConstantSpaces(buf,25,29, 1500, 3, 5,"i",7000);
//        buf=performSinusWithRandomDureeWithConstantSpaces(buf,17,18, 1500, 2, 4,"q",7000);
//      //IIF1
//        buf = "T150 V0 :CON(7, 80) ";
//        buf+=performSinusWithRandomDureeWithConstantSpaces(buf,33,25, 4000, 3, 6,"i",12000);
//        buf=performSinusWithRandomDureeWithConstantSpaces(buf,25,40, 4000, 3, 5,"s",12000);
//        buf=performSinusWithRandomDureeWithConstantSpaces(buf,17,25, 4000, 2, 4,"q",12000);
//      //IIIF1
//                                buf = "T150 V0 :CON(7, 100) ";
//                                buf+=performSinusWithRandomDuree(buf,45,15, 400, 3, 4,5500);
//                                buf+=performSinusWithRandomDuree(buf,29,20, 500, 3, 4,5500);
//                                buf+=performSinusWithRandomDuree(buf,19,20, 900, 2, 4,5500);
//                                buf+=performSinusWithRandomDuree(buf,13,20, 1200, 2, 4,5500);
//-------------------------------------------------------
//*******************************************************
    startTime = System.nanoTime();
      player.play(buf);
    endTime = System.nanoTime();
    timeElapsed = endTime - startTime;
    System.out.println("Execution time in milliseconds: " + timeElapsed / 1000000);
   }
```

```
   // Note letter (C, D, E, F, G, A, or B), accidental (#, b, natural), and octave
//• Rest is specified with an R • Duration: w, h, q, i, s, t, x, o (whole, half, etc. down to 128th)
//                                    4  8  16 32
   // pppp(1) ppp(4)   pp  p(24)  mp(55)  mf(63)  f(94)  ff(120)  fff(126)  ffff
}
```

2- cSound source code example for Windbells and the Lonely Flute

```
<CsoundSynthesizer>
<CsOptions>
-o dac
</CsOptions>
<CsInstruments>
/*
* Ali Rıza SARAL
* 20240202
*/

sr = 44100
kr = 4410
ksmps = 10
nchnls = 2
0dbfs = 1

giArr[] fillarray  24,25,26,27,28,29,30,31,32,33,34,35,                                  ;32.70;C1, C1#, D1, D1#, E1, F1, F1#, G1, G1#, A1, A1#, B1
                   36,37,38,39,40,41,42,43,44,45,46,47,                                  ;65.41;C2
                   48,49,50,51,52,53,54,55,56,57,58,59,                                  ;130.81;C3
                   60,61,62,63,64,65,66,67,68,69,70,71,                 ;central261.63;C4
                   72,73,74,75,76,77,78,79,80,81,82,83,                                  ;523.25;C5
                   84,85,86,87,88,89,90,91,92,93,94,95,                                  ;1046.50;C6
                   96,97,98,99,100,101,102,103,104,105,106,107,                          ;2093.00;C7
                   108,109,110,111,112,113,114,115,116,117,118,119,   ;4186.01;C8
                   120,121,122,123,124,125,126,127,128,129,130,131   ;8372.02;C9
                   ;107 MIDI codes

instr setPitches
    icountPitch= 108
    indx = 0
```

```
    until indx == lenarray(giArr) do
            giArr[indx] = mtof:i(giArr[indx])
            printf_i  "setPitches giArr['%d'] = '%d'\n",1,indx , giArr[indx]
            indx += 1
    od
endin

;initialize the global audio variables
gaBusL init 0
gaBusR init 0

opcode AddSynth,a,i[]i[]iooo
 /* iFqs[], iAmps[]: arrays with frequency ratios and amplitude multipliers
 iBasFreq: base frequency (hz)
 iPtlIndex: partial index (first partial = index 0)
 iFreqDev, iAmpDev: maximum frequency (cent) and amplitude (db) deviation */
 iFqs[], iAmps[], iBasFreq, iPtlIndx, iFreqDev, iAmpDev xin
 iFreq = iBasFreq * iFqs[iPtlIndx] * cent(rnd31:i(iFreqDev,0))
 iAmp = iAmps[iPtlIndx] * ampdb(rnd31:i(iAmpDev,0))
 aPartial poscil iAmp, iFreq
 if iPtlIndx < lenarray(iFqs)-1 then
  aPartial += AddSynth(iFqs,iAmps,iBasFreq,iPtlIndx+1,iFreqDev,iAmpDev)
 endif
 xout aPartial
endop

;frequency and amplitude multipliers for 11 partials of Risset's bell
giFqs[] fillarray  .56, .563, .92, .923, 1.19, 1.7, 2, 2.74, 3, 3.74, 4.07
giAmps[] fillarray 1, 2/3, 1, 1.8, 8/3, 5/3, 1.46, 4/3, 4/3, 1, 4/3

opcode AdditiveSynthRISSET, a, iiiii ;inharmonic additive synthesis
    iFreq, iDur, iVol, ifqdev, iampdev  xin
    ibasfreq = iFreq
    iamp = iVol
    ifqdev = ifqdev ;maximum freq deviation in cents
    iampdev = iampdev ;maximum amp deviation in dB
    aRisset AddSynth giFqs, giAmps, ibasfreq, 0, ifqdev, iampdev
    aRisset *= transeg:a(0, .01, 0, iamp/10, p3-.01, -10, 0)
    aout = aRisset
```

```
	xout aout
endop

;make the seed for random values each time different
seed 0
opcode RandomFreqBlock, S, iiiiii
	iPitchCount, iFloorNdx, iCeilingNdx, iDureeFloor, iDureeCeiling, iSpace xin
	puts "\nRandomFreqBlock has worked from here",1
	printf_i "\nRandomFreqBlock is called with iPitchCount = %d, iFloorNdx = %d, iCeilingNdx = %d, iDureeFloor = %d, iDureeCeiling = %d,
iSpace = %d\n",1,iPitchCount, iFloorNdx, iCeilingNdx, iDureeFloor, iDureeCeiling, iSpace

	loop:
		iDur random iDureeFloor, iDureeCeiling
		;iDur = iDuree
		timout 0, iDur, makenote
	reinit loop
	makenote:
		indx random iFloorNdx-24, iCeilingNdx-24
		iFreq = giArr[indx]
		iSign random 0, 1
		if iSign < 0.5 then
			iSign = -1
			endif
		if iFreq < 220 then
			iAlter random 1, 5
			endif
		if iFreq > 220 && iFreq < 440 then
			iAlter random 10, 20
			endif
		if iFreq > 440 && iFreq < 880 then
			iAlter random 25, 50
			endif
		if iFreq > 880 && iFreq < 1760 then
			iAlter random 50, 100
			endif
		if iFreq > 1760 && iFreq < 3520 then
			iAlter random 100, 200
			endif
		if iFreq > 3520 && iFreq < 7040 then
```

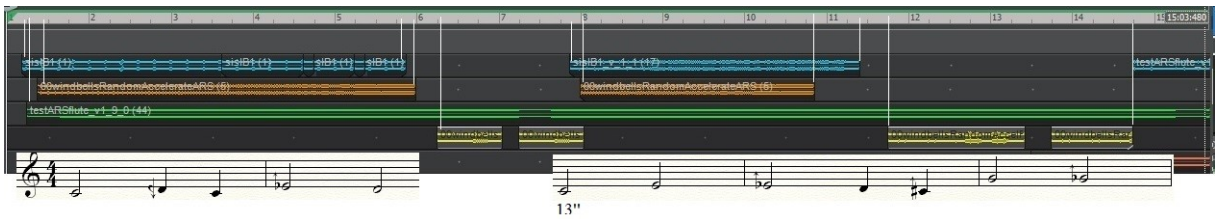
```
                iAlter random 200, 400
                endif
        if iFreq > 7040 && iFreq < 20000 then
                iAlter random 400, 800
                endif
        iFreq += iSign * iAlter
        ;iFreq += iSign
        iRest random 0, iSpace
        iVol random -6, -1; dB
        if iRest > 0.00007 then
                iFreq = 1
                iDur=0.5
                iVol = 0
                endif

        iPan random 0, 1; random panning for each signal
        aSin oscil3 ampdb(iVol), iFreq, 1
        iampdev random -0, -50
        ifqdev random 300, 700
        aSin += AdditiveSynthRISSET(iFreq, iDur, iVol, ifqdev, iampdev)
        ;aEnv transeg 1, iDur, -10, 0; env in a-rate is cleaner
        aEnv transeg 1, iDur, 0.01, 0; env in a-rate is cleaner
        aAdd = aSin * aEnv
        aL, aR pan2 aAdd, iPan
        gaBusL = gaBusL + aL; add to the global audio signals
        gaBusR = gaBusR + aR

    xout "RandomFreqBlock finished"
endop

instr performGeneric
    puts "\nperform begins here",1
    iPitchCount = 50
    iFloorNdx = P4
    iCeilingNdx = P5
    iDureeFloor = P6
    iDureeCeiling = P7
    iSpace = P8
    Sret RandomFreqBlock, iPitchCount, iFloorNdx, iCeilingNdx, iDureeFloor, iDureeCeiling, iSpace
```

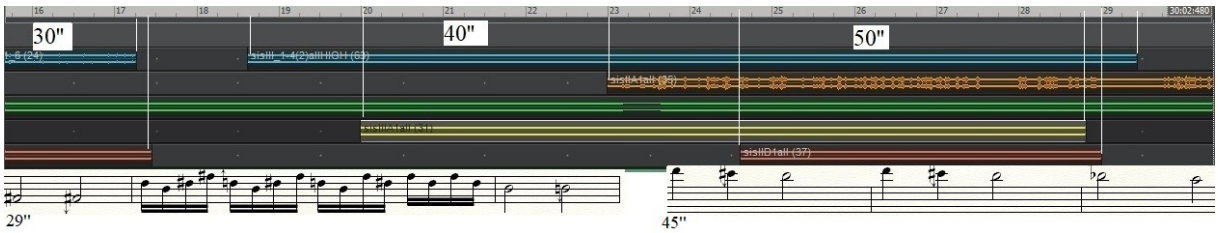
```
      ;printf_i  "\nreturned var Sa2 = '%s'\n",1, Sa2
endin

; Instrument #1.
instr 1
  ; Print out the value from timek every half-second.
  k1 timek
  printks "performance time = %3.1f seconds\\n", 0.5, k1/kr
endin

instr 5; reverb of gaBus and output
      aL, aR freeverb gaBusL, gaBusR, 0.00000001, 0.000000005    ;.1, .2   ;0.8 0.5
      outs aL, aR
endin

instr 100; clear global audios at the end
      clear gaBusL, gaBusR
endin
</CsInstruments>
<CsScore>
f 1 0 1024 10 1 .5 .3 .1
i "setPitches"
;i "performGeneric" 0 10 103 105 0.005 0.01 0.0001;2500 3500
i "performGeneric" 0 80 103 105 0.005 0.01 0.0007 ;2500 3500
i 1 0 80
i 5 0 80
i 100 0 80

</CsScore>
</CsoundSynthesizer>
;example by Ali Riza SARAL
```

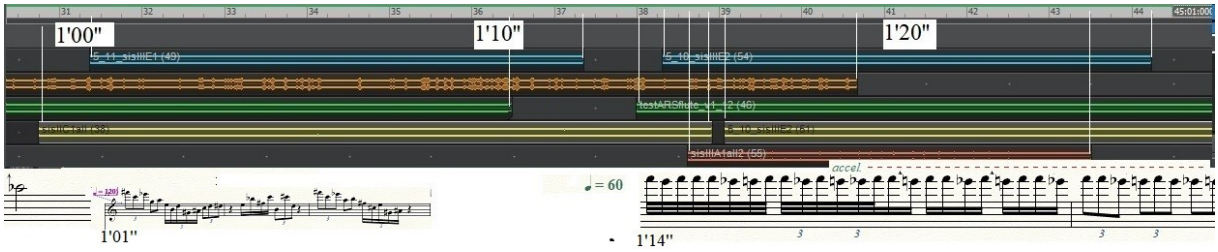
WINDBELLS AND THE LONELY FLUTE

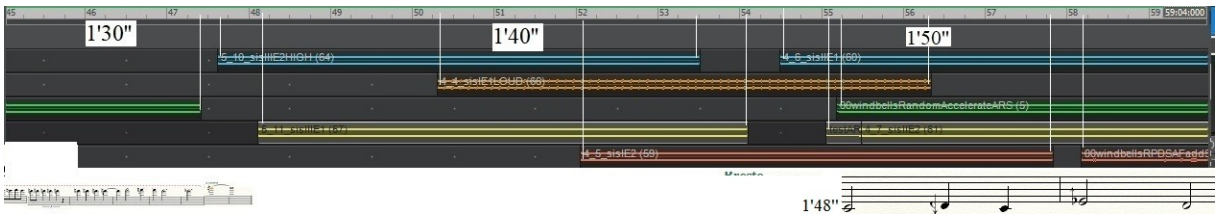
2024 April

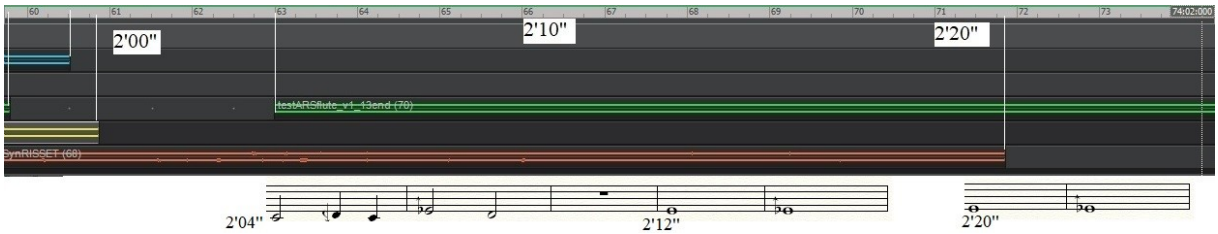
Ali Rıza SARAL







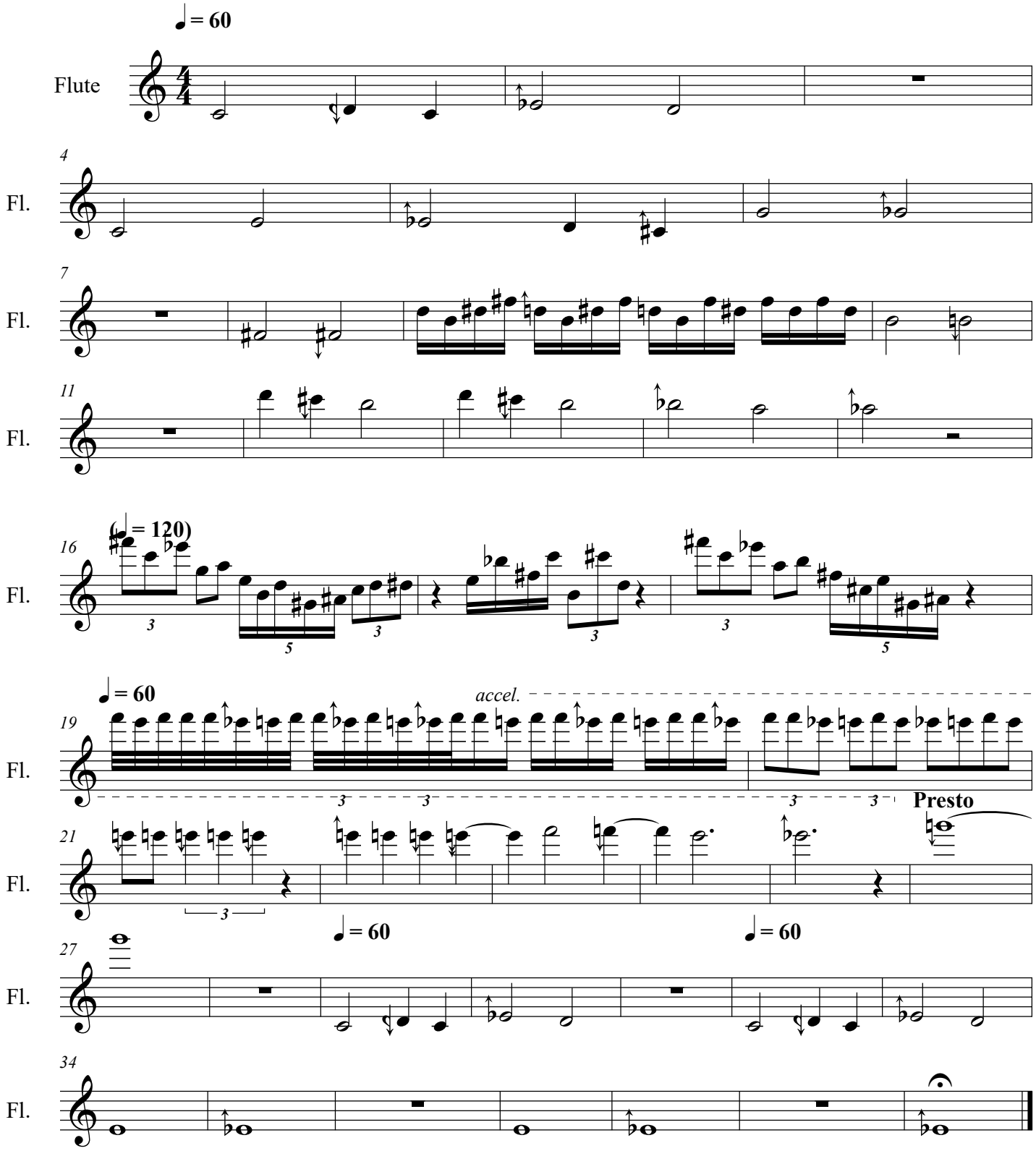




Note: Flute is the green track. The other tracks are computer generated algorithms edited by hand.

Full Score

testARS



©